vi

Pour les articles homonymes, voir VI.

vi est un éditeur de texte en mode texte plein écran écrit par Bill Joy en 1976 sur une des premières versions de la distribution Unix BSD.

Il est présent d'office sur la majorité des systèmes Unix actuels, souvent sous la forme d'un clone du logiciel *vi* originel.

Sommaire

1 Origine du nom
2 Principe de fonctionnement
3 Évolution
4 Clones et dérivés
5 Débuter avec vi
5.1 Commandes et interface jugées obscures
5.2 Comportement modal
6 Fonctionnalités de vi
6.1 Macros et abréviations
6.2 Options
6.3 Fichier de configuration
7 Exemples de commandes
7.1 Exemples de commandes simples
7.2 Exemples de commandes avancées
8 Notes et références
9 Voir aussi
9.1 Articles connexes
9.2 Bibliographie
9.3 Liens externes

Origine du nom [modifier le code]

Le nom **vi** provient de l'abréviation la plus courte possible (c'est-à-dire sans ambiguïté) de la commande `visual` de l'éditeur de texte **ex**. Cette commande passe ex du mode *ligne par ligne* en mode *plein écran*. Ainsi, vi était au départ conçu comme une interface visuelle (en anglais, *Visual Interface*) rajoutée à l'éditeur ex (lui-même basé sur ed). ex est toujours disponible dans vi en appuyant sur `:` en mode commande. En pratique, un unique programme est démarré lors de l'invocation de *vi* ou *ex*, son comportement change selon le nom avec lequel il est appelé.

Le nom `vi` correspondant à la fois à des initiales et au nom de son fichier d'installation, il est usuellement prononcé en énonçant les deux lettres en anglais, c'est-à-dire *« vi-aille »*, [vi: aɪ], plutôt que comme un mot à part entière [vi]. L'analogie de ce nom avec le chiffre romain *VI* (six) est fortuite.

Principe de fonctionnement [modifier le code]

vi est un éditeur modal, c'est-à-dire que la signification des boutons et des touches change selon le mode dans lequel il se trouve.

En mode *insertion*, le texte tapé est inséré dans le document. Appuyer sur la touche *Echap* depuis le mode insertion permet de passer dans le mode *commande*, dans lequel les touches correspondent à des déplacements du curseur ou à des fonctions d'édition. Par exemple, `j` descend le curseur d'une ligne, `x` efface le caractère sous le curseur (la position « sous le curseur » peut désigner la droite du curseur si ce dernier se place entre les caractères, au-dessus du caractère de soulignement ou sous le bloc rectangulaire, selon la manière dont le terminal représente le curseur).

Les touches tapées en mode commande ne sont pas insérées dans le texte, ce qui est une cause fréquente de confusion pour les utilisateurs débutants avec vi.

En mode *commande*, de nombreuses opérations peuvent être effectuées en série avec des séquences de touches simples, sans qu'il soit nécessaire de maintenir les touches *Alt* ou *Ctrl* enfoncées. Les opérations les plus élaborées sont composées d'opérations plus primaires. Par exemple, `d3w` efface trois mots (`d` pour *delete* (effacer) et `w` pour *word* (mot), `c2fa` change (*change*) le texte jusqu'à ce qu'il trouve (*find*) le second (*2*) *a*. Pour les utilisateurs expérimentés, cela permet de travailler très efficacement. Cela permet également à l'utilisateur de conserver en permanence ses mains sur le clavier.

Українська
Tiếng Việt
中文
Modifier les liens

Les premières versions de vi ne donnaient aucune indication sur le mode dans lequel elles se trouvaient. Il était fréquent que les utilisateurs tapent machinalement sur la touche *Echap* pour s'assurer que l'éditeur était bien dans le mode commande (vi émet un signal sonore s'il est déjà dans ce mode). Les versions plus modernes de vi indiquent le mode dans une barre d'état ou graphiquement (par exemple la forme ou la couleur du curseur). Des implémentations graphiques de vi (par exemple GVim) supportent aussi l'utilisation de la souris et des menus pour accéder aux fonctions d'édition.

Évolution [modifier le code]

vi est devenu de facto l'éditeur standard d'unix et il a été l'éditeur favori de nombreux hackers jusqu'à l'arrivée d'Emacs en 1984. Il est à noter qu'Emacs est bien plus qu'un simple éditeur de texte et est pourtant souvent mis en concurrence avec vi. À ce jour (2011), vi ou l'un de ses clones peut être trouvé dans presque toutes les installations de Unix. La Single UNIX Specification (plus particulièrement l'« IEEE standard 1003.2, Part 2: Shell and utilities ») inclut vi. Ainsi, tout système se conformant à cette spécification intègre vi.

vi est encore largement utilisé par les utilisateurs des différentes variantes d'Unix. Il démarre plus vite que les versions lourdes de l'« éditeur » Emacs et utilise moins de mémoire. Conséquemment, mêmes des fans d'Emacs l'utilisent comme éditeur pour le courrier électronique ou pour de petites éditions.

Lors de la création d'une disquette de récupération (*rescue disk*, pour les systèmes dont le disque dur ne fonctionne plus correctement), vi est bien souvent choisi comme éditeur, en raison de sa compacité (la place est très limitée sur les disquettes) et du fait que la majorité des gens effectuant des opérations de récupération sont capables de l'utiliser.

vi et Emacs sont les éternels belligérants de la guerre des éditeurs.

Clones et dérivés [modifier le code]

De nombreux éditeurs de texte basés sur vi existent. Les clones sont des mises en œuvre, libres ou non, de vi, c'est-à-dire qu'ils respectent théoriquement le jeu de commandes standard du vi de POSIX. Toutefois, la majorité de ces clones supportent également des fonctionnalités supplémentaires. Les dérivés sont des programmes, éditeurs de texte ou non, qui ont conservé le principe de fonctionnement de vi (en partie ou complètement) pour leur propre fonctionnement.

- vi est un portage vers les systèmes Unix modernes du vi classique de la BSD 3.7. Cette implémentation, qui s'appuie sur le code de ed, est libre au sens BSD depuis janvier 2002[1].
- nvi est une implémentation de l'éditeur de texte ex/vi distribuée dans la Fourth Berkeley Software Distribution (4BSD). Il s'agit de la version de vi qui est fournie avec toutes les distributions BSD open source. Elle possède de nombreuses améliorations par rapport au vi classique, notamment l'historique et l'édition des commandes, le complètement des noms de fichiers, l'édition de plusieurs tampons, le multi-fenêtrage (y compris l'édition d'un même tampon dans plusieurs fenêtres), etc. À part ces améliorations, son interface utilisateur est à peu de choses près identique à celle du vi classique. Cette implémentation est très compacte, rapide, et se fond très naturellement dans l'environnement Unix.
- Elvis est un clone libre de vi pour Unix et d'autres systèmes d'exploitation. Il possède des fonctionnalités supplémentaires, telles que la coloration syntaxique, ou une interface graphique pour différents systèmes.
- Vim, pour « Vi IMproved », est un clone de vi, moderne et extensible. Il est généralement fourni comme implémentation de vi dans les distributions Linux.
- svicc est un petit clone de Vi pour Commodore (64)[2]
- bvi « Binary VI » est un éditeur pour les fichiers binaires, basé sur l'éditeur de texte vi[3].
- VILE, pour « VI Like Emacs », est un éditeur de texte qui tente de réconcilier les utilisateurs de vi et de emacs en gardant ce qu'il estime être le meilleur des deux programmes.

Une liste plus complète peut être trouvée sur dans les liens externes ci-après.

Débuter avec vi [modifier le code]

Les utilisateurs, débutant avec vi, sont souvent confrontés à des difficultés, d'une part à cause des raccourcis utilisés pour chacune des commandes, ensuite parce que l'effet de ces raccourcis change selon le mode dans lequel se trouve vi.

Un moyen efficace de débuter avec vi est de suivre d'un bout à l'autre un tutoriel expliquant les commandes les plus simples. La plupart des systèmes fournissant une implémentation de vi contiennent également un tutoriel pour vi.

En comprenant son principe de fonctionnement avec quelques explications et des exemples simples, la majorité des utilisateurs peuvent parvenir à leurs fins avec vi. Toutefois, l'aisance avec vi est généralement considérée comme plus longue à venir qu'avec les autres éditeurs avec lequel il est souvent comparé. Les défenseurs de vi affirment que cette difficulté initiale est largement compensée par l'efficacité de l'éditeur une fois que l'on est à l'aise.

Commandes et interface jugées obscures [modifier le code]

Les débutants sont souvent déroutés par les commandes de vi. Ces dernières sont bien souvent réduites à une seule et unique lettre, et bien difficiles à mémoriser au premier abord, d'autant que l'éditeur présente différentes perspectives (lecture, insertion, open mode, etc.) qui ne sont pas clairement distinguées et la transition entre ces modes passe souvent inaperçue.

La plupart des commandes de vi sont choisies de façon à :

- limiter la frappe nécessaire. Les modificateurs tels que *Ctrl*, *Maj* ou *Alt* sont utilisés avec la plus grande parcimonie ;
- limiter les mouvements des doigts et des mains sur le clavier. Par exemple, en mode commande, les touches *h*, *j*, *k* et *l* permettent de déplacer le curseur. Comme il s'agit d'une des fonctions les plus importantes, les touches qui lui sont affectées sont celles que l'utilisateur a immédiatement sous les doigts ;
- faciliter les moyens mnémotechniques pour retenir leur(s) effet(s). Il faut toutefois garder à l'esprit que vi a été écrit par des programmeurs anglophones. Par exemple, en mode commande, `d` permet d'effacer (**d**elete), `i` passe en mode **i**nsertion, `w` avance le curseur d'un mot (**w**ord). En combinant une commande d'édition (par exemple effacer : `d`) et une commande de mouvement (par exemple avancer d'un mot : `w`), on obtient la séquence `dw`, qui permet d'effacer un mot. D'une manière similaire, la commande `d3w` efface trois mots.

Comportement modal [modifier le code]

Lorsque l'on débute avec vi, il est primordial de comprendre que vi possède plusieurs modes de fonctionnement. Les deux principaux modes sont le mode *commande* (dans lequel vi démarre) et le mode *insertion*. Les commandes `i` (**i**nsertion) ou `a` (**a**jout) du mode commande permettent de passer en mode insertion, tandis que la touche *Echap* permet de quitter le mode insertion et retourner au mode commande.

vi possède également un autre mode de fonctionnement, qui offre la possibilité de définir des macro-commandes en intégrant un véritable langage de programmation pour automatiser des tâches d'édition de texte.

Il est aussi possible d'appeler vi avec des arguments afin qu'il exécute automatiquement des commandes lors de l'ouverture d'un fichier. Par exemple, la ligne de commande :

```
vi -c "1,%s/Deux/Trois/g|:wq" FICHIER
```

lance vi, qui ouvre le fichier "FICHIER", puis remplace tous les "Deux" par des "Trois" dans le fichier, et enfin enregistre le fichier et quitte.

Fonctionnalités de vi [modifier le code]

Macros et abréviations [modifier le code]

vi permet de définir des macro-commandes et des abréviations.

Les macros permettent de remplacer une séquence de touches par une autre, ce qui permet de définir de nouvelles commandes. Par exemple,

```
:map v 10dd
```

crée la commande `v` du mode commande, qui supprime des blocs de 10 lignes.

Les abréviations permettent d'éviter de taper une séquence de lettres plutôt longue. Par exemple :

```
:ab GNU gnu is not unix
```

crée l'abréviation *GNU*, qui en mode insertion est remplacée par *gnu is not unix* dès que suffisamment de caractères sont tapés pour lever toute ambiguïté (les abréviations sont généralement étendues quand un espace est inséré pour indiquer la fin du mot).

Options [modifier le code]

Le comportement de vi et de ses commandes peut être ajusté grâce à ce que vi appelle des « options ». La commande `:set` permet de les consulter et de les modifier. Certaines options sont booléennes, c'est-à-dire qu'elles peuvent prendre deux valeurs seulement : *vrai* ou *faux*, ou plus exactement *activée* et *désactivée*. Par exemple :

```
:set number
```

que l'on peut abréger par

```
:set nu
```

affiche le numéro de chaque ligne.

```
:set nonumber
:set nonu
```

sont équivalentes et permettent de désactiver cette option.

Parmi les options booléennes, on peut trouver (liste non exhaustive) :

Commande	Effet
:set ignorecase	ignore la casse lors de la recherche de chaînes de caractères
:set readonly	interdit la modification du fichier ouvert
:set autoindent	indente automatiquement le code lors de l'édition
:set ruler	indique le numéro de ligne et de colonne sur lesquelles se trouve le curseur
:set showmode	montre le mode dans lequel se trouve vi

Les autres options prennent une valeur. Par exemple :

```
:set tags=./tags
```

indique à vi où chercher le fichier de marqueurs.

L'ensemble de toutes les options disponibles peut être affiché avec

```
:set all
```

Fichier de configuration [modifier le code]

L'ensemble des options de vi (ainsi que les macros et les abréviations) peuvent être fixées pour chaque session d'édition en les ajoutant dans le fichier *~/.exrc*. Ce fichier est lu automatiquement au démarrage de vi, et les commandes qu'il contient sont exécutées comme si elles étaient tapées par l'utilisateur (le `:` initial est inutile). Exemple :

```
set nu
map @ dd
ab sig Mr. Robert Le NomBienLong
```

Exemples de commandes [modifier le code]

Exemples de commandes simples [modifier le code]

Le tableau qui suit donne quelques commandes basiques de l'éditeur vi, nécessaires aux tâches d'édition les plus simples. Pour entrer ses commandes, il faut préalablement quitter le mode éditeur en appuyant sur la touche *Echap*.

Commande	Effet
#*commande*	exécute # fois la *commande* (par exemple `5dd` efface 5 lignes)
!*commande*	exécute une commande shell (par exemple `:!gcc -o hello hello.c`)
>> et <<	ajoute et retire une tabulation en début de ligne respectivement
`/toto` et `?toto`	recherche les occurrences de la chaîne de caractères `toto` après le curseur et avant le curseur respectivement (`n` va à l'occurrence suivante et `N` va à l'occurrence précédente)
`:#`	place le curseur à la ligne #
`:$`	place le curseur à la dernière ligne
`:.=`	renvoie le numéro de la ligne courante
`0`	déplace le curseur au début de la ligne
`$`	déplace le curseur à la fin de la ligne
`h` et `l`	déplace le curseur d'un caractère à gauche et à droite respectivement

j et k	déplace le curseur sur la ligne suivante et précédente respectivement
gg	déplace le curseur au début du fichier
G	déplace le curseur à la fin du fichier
H et L	déplace le curseur en haut et en bas de l'écran respectivement
a	insère du texte après le curseur
A	insère du texte à la fin de la ligne courante
c	efface un certain nombre de caractères en fonction de la sous-commande et passe en mode insertion (par exemple cl permet de changer le caractère courant, cw la fin du mot courant)
dd	supprime la ligne courante et la copie dans le buffer
dw	supprime le mot après le curseur et le copie dans le buffer
dW	supprime la chaîne de caractères après le curseur jusqu'au prochain caractère blanc
D	supprime la fin de la ligne à partir du curseur
i	insère du texte avant le curseur
o	ouvre une nouvelle ligne en dessous de la ligne courante
O	ouvre une nouvelle ligne au-dessus de la ligne courante
p	insère le contenu du buffer après le curseur (ou après la ligne courante si le buffer contient des lignes)
P	insère le contenu du buffer avant le curseur (ou avant la ligne courante si le buffer contient des lignes)
:q	quitte vi
:q!	quitte vi même si les modifications n'ont pas été enregistrées
r	remplace le caractère sous le curseur
u	annule les effets de la commande précédente
w	place le curseur sur le premier caractère du mot suivant
:w	enregistre le fichier courant
:wq ou :x ou ZZ	enregistre le fichier courant et quitte
x	supprime le caractère sous le curseur
yy	copie la ligne courante dans le buffer (p insère la ligne copiée après la ligne courante)
yw	copie le mot après le curseur dans le buffer (p insère le mot copié après le curseur)

Exemples de commandes avancées [modifier le code]

Les commandes suivantes montrent ce que permet vi quand un utilisateur cherche des commandes d'édition plus pointues.

Commande	Effet
"ryy	mémorise la ligne courante dans le registre *"r*
"rp	insère le contenu du registre *"r* après la ligne courante ou après le curseur
:#1,#2t#3	copie les lignes #1 à #2 après la ligne #3
:#1,#2m#3	déplace les lignes #1 à #2 après la ligne #3
:#1,#2s/toto/tutu/g	remplace aux lignes #1 à #2 toutes les occurrences de la chaîne de caractères toto par tutu
:%s/toto/tutu/g	remplace à chaque ligne toutes les occurrences de la chaîne de caractères toto par tutu
:%s%to/to%tu/tu%g	remplace à chaque ligne toutes les occurrences de la chaîne de caractères to/to par tu/tu (cela montre que l'on peut utiliser un autre caractère séparateur que / — ici %)
:s/.*/\U&/	passe la ligne courante en majuscule
:%s/^toto//	efface à chaque ligne la première occurrence de la chaîne de caractères toto si elle est en début de ligne
:#1,#2g/toto/d	efface parmi les lignes #1 à #2 les lignes contenant la chaîne de caractères toto
:g!/toto/d	efface les lignes ne contenant pas la chaîne de caractères toto

`:g/^[\.]*$/d`	supprime les lignes vides
`:#1,#2g/toto/s/tutu/titi/gc`	remplace après confirmations aux lignes #1 à #2 contenant la chaîne de caractères `toto` toutes les occurrences de la chaîne de caractères `tutu` par `titi`
`:r toto.txt`	insère le contenu du fichier `toto.txt`
`:r!ls -l`	insère le résultat de la commande `ls -l` dans le fichier courant
`:#1,#2w toto.txt`	enregistre les lignes #1 à #2 dans le fichier `toto.txt`
`:e!`	ouvre à nouveau le fichier courant (sert par exemple si le fichier a été modifié entre temps)

Notes et références [modifier le code]

1. ↑ ex-vi sur SourceForge.net
2. ↑ svicc sur floodgap.com
3. ↑ bvi sur SourceForge.net

Voir aussi [modifier le code]

Articles connexes [modifier le code]

- Vim, « Vi IMproved » (vi amélioré) : un clone de vi plus puissant et plus simple que l'original ;
- Emacs - concurrent traditionnel de Vi dans le cadre de la guerre d'éditeurs.
- ed ;
- nvi, implémentation libre (BSD) de vi ;
- Elvis, autre implémentation de vi, plus puissante que l'originale ;
- Guerre d'éditeurs.

Bibliographie [modifier le code]

- (en) Linda Lamb, Arnold Robbins, *Learning the vi Editor, Sixth Edition*, O'Reilly, 1998 (ISBN 978-1-56592-426-0)

Liens externes [modifier le code]

- (fr) Complément de cours en français ;
- (fr) Page de manuel de la commande
- (en) vi lovers home page - La page des amoureux de vi ;

v · d · m	Commandes Unix
Aide	arch · apropos · cal · date · help · file · history · info · man · type · uname · whatis
Environnement utilisateur	env · dmesg · finger · halt · id · init · logname · mesg · passwd · reboot · shutdown · su · sudo · uptime · w · wall · who · whoami · write
Gestion de processus	at · bg · chroot · crontab · exit · fg · kill · lp · lpr · nice · nohup · pkill · ps · pstree · sleep · time · top
Manipulation de texte	awk (gawk) · banner · bash · bc · cat · clear · cut · ed · head · join · nl · read · sed · sort · strings · tail · tee · tr · uniq · **vi** · vim · wc
Programmation shell	alias · basename · dirname · echo · expr · false · printf · set · test · true · unset · wait
Recherche	find · grep · locate · whereis · which
Réseau	ethtool · ftp · host · ifconfig · mail · mii-tool · nc · netstat · ping · rlogin · ssh · sftp · telnet · traceroute
Système de fichiers	cd · chgrp · chmod · chown · cp · dd · dir · du · df · file · less · ln · ls · lsof · mkdir · more · mount · mv · pwd · rm · rmdir · split · touch · umask · umount

Portail de l'informatique

Catégories : Éditeur de texte | Logiciel pour Unix

Dernière modification de cette page le 22 mars 2014 à 23:03.

Droit d'auteur : les textes sont disponibles sous licence Creative Commons paternité partage à l'identique ; d'autres conditions peuvent s'appliquer. Voyez les conditions d'utilisation pour plus de détails, ainsi que les crédits graphiques. En cas de réutilisation des textes de cette page, voyez comment citer les auteurs et mentionner la licence.
Wikipedia® est une marque déposée de la Wikimedia Foundation, Inc., organisation de bienfaisance régie par le paragraphe 501(c)(3) du code fiscal des États-Unis.

Politique de confidentialité À propos de Wikipédia Avertissements Développeurs Version mobile